Technical Report 392

(12) LEVEL II

AN EXPLORATORY STUDY OF THE COGNITIVE STRUCTURES UNDERLYING THE COMPREHENSION OF SOFTWARE DESIGN PROBLEMS

Michael E. Atwood, Althea A. Turner, and H. Rudy Ramsey
Science Applications, Inc.

and

Jean Nichols Hooper
Army Research Institute

HUMAN FACTORS TECHNICAL AREA

U. S. Army
Research Institute for the Behavioral and Social Sciences

July 1979

U. S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency under the Jurisdiction of the Deputy Chief of Staff for Personnel

JOSEPH ZEIDNER
Technical Director

WILLIAM L. HAUSER
Colonel, U S Army
Commander

Research accomplished
under contract to the Department of the Army

Science Applications, Inc.

NOTICES

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: Technical Report 392 (18) ARI	2. GOVT ACCESSION NO.: TR-392
3. RECIPIENT'S CATALOG NUMBER: (9)	
4. TITLE (and Subtitle): (6) AN EXPLORATORY STUDY OF THE COGNITIVE STRUCTURES UNDERLYING THE COMPREHENSION OF SOFTWARE DESIGN PROBLEMS	5. TYPE OF REPORT & PERIOD COVERED: Technical Report 3 Oct 1977 - 3 Dec 1978
	6. PERFORMING ORG. REPORT NUMBER: (14) SA1-79-100-DEN
7. AUTHOR(s): (10) Michael E. Atwood, Althea A. Turner, H. Rudy Ramsey, and Jean Nichols Hooper (ARI)	8. CONTRACT OR GRANT NUMBER(s): (15) DAHC19-78-C-0005
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Science Applications, Inc. 7935 E. Prentice Avenue Englewood, CO 80111	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: (16) 2Q762725A778
11. CONTROLLING OFFICE NAME AND ADDRESS: U.S. Army Research Institute for the Behavioral and Social Sciences (PERI-OS) 5001 Eisenhower Avenue, Alexandria, VA 22333 (11)	12. REPORT DATE: July 1978
	13. NUMBER OF PAGES: 53
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): -- (12) 62 p.	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

--

18. SUPPLEMENTARY NOTES

Monitored technically by Jean Nichols Hooper and Edgar M. Johnson, Human Factors Technical Area, ARI.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer programming
Computer program documentation
Specifications
Memory (psychology)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An experiment was conducted to evaluate a framework for the study of software complexity and comprehension. Basic to this framework is the concept that a person's knowledge of, and experience with, software design affects that person's ability to comprehend a software problem and its potential solutions. Past research on software complexity and comprehensibility has largely been based on the assumption that complexity is a function of surface properties, such as variable names and flow of control. Such measures, however, ignore the effects of experience. → next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

392 878

20.

Research on expert-novice differences in problems solving suggests that experts possess a large number of previously developed knowledge structures, or schemata, that can be used to understand or solve the current problem. Research on text comprehension provides theoretical concepts and experimental paradigms that are useful in determining the structure and content of these experience-related schemata.

An experiment examined the knowledge structures used by participants, at differing levels of experience, in comprehending software system specifications. Six participants, at each of five levels, studied a software system specification and then summarized both the presented specification and the probable form of the corresponding software design. The results indicate that software designers use previously learned schemata in understanding a software design problem and in actually constructing a design and that these schemata differ as a function of experience. In addition, the structure and content of these schemata were investigated. It is suggested that by determining the structure and content of such schemata, software complexity and comprehensibility can be considered in a more meaningful manner.

Technical Report 392

AN EXPLORATORY STUDY OF THE COGNITIVE STRUCTURES UNDERLYING THE COMPREHENSION OF SOFTWARE DESIGN PROBLEMS

Michael E. Atwood, Althea A. Turner, and H. Rudy Ramsey
Science Applications, Inc.

and

Jean Nichols Hooper
Army Research Institute

Raymond C. Sidorsky, Team Chief

Submitted by:
Edgar M. Johnson, Chief
HUMAN FACTORS TECHNICAL AREA

Approved by:

Frank J. Harris, Acting Director
ORGANIZATIONS AND SYSTEMS RESEARCH LABORATORY

U.S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES
5001 Eisenhower Avenue, Alexandria, Virginia 22333

Office, Deputy Chief of Staff for Personnel
Department of the Army

July 1979

Army Project Number
2Q762725A778

Information Processing
and Display

FOREWORD

The Human Factors Technical Area of the Army Research Institute (ARI) is concerned with human resource demands of increasingly complex battlefield systems used to acquire, transmit, process, disseminate, and utilize information. This increased complexity places great demands upon the operator interacting with the machine system. Research in this area focuses on human performance problems related to interactions within command and control centers as well as issues of system development. It is concerned with such areas as software development, topographic products and procedures, tactical symbology, user-oriented systems, information management, staff operations and procedures, and sensor systems integration and utilization.

One area of special interest involves the development of computer software to support automated battlefield systems. Software development is a costly, unreliable, not well understood process. The research reported here applied a theoretical framework based on representation of text in memory to the production of software design and specification summaries. The findings verified the prediction that experience is a determinant of the knowledge structures employed in representing design information. The research is part of a larger effort to develop a conceptualization of the programming process and identify behavioral bottlenecks in software development. Efforts in this area are directed at improving accuracy and productivity in programming through the design of procedures, languages, and methods to enhance programmer performance.

Research in the area of human factors in software development is conducted as an in-house effort augmented contractually by organizations selected as having unique capabilities and facilities; in this case, Science Applications, Incorporated, under Contract DAHC19-78-C-0005. The effort is responsive to requirements of Army Project 2Q762725A778, and to general requirements expressed by members of the Integrated Software Research and Development Working Group (ISRAD).

JOSEPH ZEIDNER
Technical Director

AN EXPLORATORY STUDY OF THE COGNITIVE STRUCTURES UNDERLYING THE COMPREHENSION OF SOFTWARE DESIGN PROBLEMS

BRIEF

Requirement:

To develop and test a theoretical framework for guiding and integrating future research on measures of software complexity and comprehensibility.

Procedure:

Research on expert-novice differences in problem solving suggests that experts differ from novices in the number and type of experience-related schemata, or memory structures, that can be applied to a current problem. These general schemata are used to develop a macrostructure, which describes the problem solver's understanding of a particular problem. Problem information that corresponds to a problem solver's schemata does not require additional processing in order to be comprehended; information that does not correspond, however, must be processed and incorporated into a macrostructure. Techniques derived from research on text comprehension were used to determine the macrostructures formed by participants at varying levels of experience to represent a software system specification. By analyzing the macrostructure information included by less experienced participants but omitted by more experienced participants, the schemata used by experienced participants were inferred.

Findings:

Software designers use previously developed schemata, or memory structures, to understand a software design problem and to construct a design. These schemata differ as a function of experience. Further, examination of the form and content of these schemata led to the conclusion that software complexity and comprehensibility can be defined in terms of such schemata.

Utilization of Findings:

The actual specification presented to a software designer can vary both in form and content. Structuring these specifications to correspond more closely to the designer's schemata would aid the designer both in comprehending these specifications and constructing the indicated design. The complexity of a software design or program should be defined in terms

of the deviations from the designer's or programmer's existing schemata. Software development procedures, techniques, and training programs should make use of available schemata and should aid the formation of relevant additional schemata.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

The production of software is primarily a human activity. Although tremendous advances in both hardware and software technology have been made in the past decade, the development of software systems remains a predominantly human activity. As a result, the ability of software development personnel to comprehend the nature of a project and its required solution affects the cost, time, and success of that project. The comprehensibility of a software project is determined, in large part, by its complexity.

In recognition of the importance of the human component, a large number of software development methods, "rules of thumb," have been proposed. These techniques and guidelines are intended to make software easier for humans to comceptualize and, therefore, to specify and proceduralize. As yet, the usefulness and validity of these methods have not been proven. However, these methods indicate an important approach to software development, for they imply that the complexity and comprehensibility of a given software project is determined by those who develop the software as well as by the nature of the project itself.

This paper describes our initial empirical and theoretical efforts at developing a coherent framework for the study of software complexity and comprehension. Basic to this framework is the idea that a person's knowledge of, and experience with, software design affects that person's ability to comprehend a software problem and its potential solutions.

There is a growing literature on software complexity and comprehension. However, we feel that the usefulness of this literature is extremely limited due to the prevalent definitions of the terms "complexity" and "comprehension". The concepts discussed in this literature differ from those we wish to investigate.

Past research on software comprehensibility and complexity has usually been based on the assumption that the complexity of a computer program is a function of such surface properties as control structures, variable names, and similar aspects. Examples of such studies can be found in Sime, Green, and Guest (1973, 1977), Newsted (1974), Shepard and Love (1977), Miller (1975), and Weissman (1973, 1974). These, and similar studies, have considered how comprehensibility is influenced by comment statements, mnemonic variable names, meaningful variable names, alternative language constructs, etc.

Although at first glance it may appear reasonable that such aspects do affect program complexity and comprehensibility, we argue that when the effects of programmer experience are ignored no general statements about complexity or comprehensibility are possible. This point is illustrated in an experiment reported by Shneiderman (1976). Shneiderman compared the comprehensibility of arithmetic and logical IF statements using participants of two different experience levels. It was found that logical IF's were more "comprehensible" than arithmetic IF's for the less experienced participants. There were no differences in "comprehensibility" for the more experienced participants.

We believe that complexity and comprehensibility are functions of an individual's experiences and abilities. Imagine that we have a group of programmers of varying levels of experience. In effect, we have a dimension labelled "experience"; at one extreme we have programmers that we characterize as "expert" and at the other extreme, we have "novices". Assume we have a single program, such as a sort routine. We present this program to all of these programmers indivdually and ask them to rate its complexity. We would expect to see a wide range of complexity ratings. The "expert" may consider this program to be "trivial", but the "novice" categorizes it as "incomprehensible". The crucial question that remains to be answered, and the one that we will address in this paper, is why this statement is true.

How do the expert and novice differ in their ability to comprehend this program?

Our answer to this question can be illustrated by continuing the above example. After examining the presented program, we would not be surprised if the experts made comments such as "this is like a Shell sort, except that ..." or "this is really a bubble sort with the addition of ...". Similarly, less experienced programmers may note that "the program appears to be interchanging values, but I'm not quite sure why or how".

The expert has seen a large number of sort routines. Further, the expert can explain how these routines work, the principal differences among alternative sort techniques, etc. The expert "knows" these things because, through experience, this programmer is able to "understand" sort routines. This "understanding" means that the expert has organized and integrated various pieces of information about sort routines and has stored this information in memory as coherent knowledge structures. Comprehension, in this case, is driven or guided by these existing knowledge structures. The absence of such knowledge structures makes comprehension by the novice much more difficult.

In this paper, we will focus on these knowledge structures and attempt to determine how these structures develop as a function of experience. If we are able to develop a sufficient understanding of these knowledge structures, then we will have a better understanding of the concepts of software complexity and comprehensibility. As a result of this understanding, it may be possible to suggest software development procedures that lead to the production of more comprehensible software, with corresponding reductions in the cost and time associated with software development.

The Effects of Expertise on Schema Selection

Complexity and comprehensibility then can only be defined and measured with respect to the level of expertise of a given programmer,

or group of programmers. In order to define and measure these concepts, therefore, we must be able to categorize what is implied by "expertise". That is, why is an expert able to perform some task or understand some problem better than a novice?

Within the past few years, cognitive psychologists have attempted to answer this question by comparing the performance of experts and novices in a variety of tasks. This research supports the conclusion that expert problem solving behavior is strongly influenced or driven by existing memory structures, or "schemata". That is, an expert has a large number of generalized plans, solution strategies, or schemata for a given type of problem. When presented with a problem the expert attempts to retrieve one of these schemata and adapt it to achieve a solution to the current problem.

The standard paradigm in this type of research is to compare the performance of experts and novices on problems that both can solve and to examine the problem solving processes used by subjects of different levels of expertise. A good example of this type of research is the study of Chase and Simon (1973). They found that highly experienced chess players differ from "good" amateur players not in their ability to execute more efficient search strategies or apply more sophisticated problem solving processes or in their ability to consider a larger number of potential moves, but rather in the experts' memory for a larger number of chess positions and the "correct", or optimal move associated with each position. In other words, experts have a large number of situation-specific schemata.

In the area of engineering thermodynamics, Bhaskar and Simon (1977) have demonstrated that there are a relatively small number of well-defined schemata. In this case, the schemata are based on the basic thermodynamics equations, which are few in number. In more complex tasks, however -- and software design appears to be such a task -- these schemata may be more complex.

Larkin (1977) has considered behavior on problems which are apparently more complex than those used by Bhaskar and Simon -- mechanics problems in physics. Again, schemata were closely tied to the standard equations that are found in physics textbooks. These schemata, however, were organized into larger schemata. In particular, these equation-based schemata were organized into "chunks" of related equations and principles by expert physicists. Rather than retrieving equation-based schemata, the experts initially retrieve one or more appropriate chunks and only then consider the individual equations involved. In essence, there is a hierarchy of schemata, ranging from extremely general to situation-specific. Larkin's novices, however, did not possess this hierarchical knowledge structure and immediately began retrieving equation-based schemata.

Notice that there is no guarantee that the individual equations retrieved by the novice are, in fact, applicable to the current problem. Each equation must be retrieved, tested for relevance, and either applied or rejected. In effect, this is a type of trial-and-error behavior. Because these schemata are chunked, however, the expert can quickly consider and accept or reject a number of equations and, in effect, filter the knowledge that need be considered.

The research reviewed in this section supports the conclusion that the problem solving performance of experts and novices differs due to the memory structures, or schemata, employed by the individual. The principal question that we have left unanswered is what these knowledge structures consist of and how they are organized.

In research on human problem solving, this question has not been directly addressed. Methodologies which could be used to determine the structure of the knowledge that guides problem solving have been developed and successfully applied in text comprehension studies. Although these studies are more concerned with the integration, storage, and retrieval of information than with the structures already in memory, these paradigms may also be useful in determining the structure of

existing knowledge. Below, we will briefly describe this research and associated paradigms and indicate how these paradigms may be applied to the question above. Following this, we present the results of an experiment directed toward differentiating between the knowledge structures utilized by experts and novices in software design.

Kintsch and van Dijk (1978) have theorized that people have several types of knowledge structures available to them when processing text. One of these is the "schema". Schemata are generic knowledge structures which specify principal elements, characteristic categories and procedures for a particular type of information. When a given schema (such as a narrative schema or a psychological report schema) is called up, its information is made available to the problem solver. Some of this information can be used to direct processing, to indicate what is relevant or which parts are obligatory, etc.

The schema can be represented as a tree structure of the hierarchically organized information which is associated with the schema's theme or topic. The terminal nodes of each branch of the schema are empty slots, each of which has a set of conditions which potential values must meet. When a situation invokes a particular schema, these slots are filled with information from the situation which meet their conditions.

An example of a schema is the narrative story schema (Kintsch & Van Dijk, 1978). This particular structure has received attention from a number of other researchers (Rumelhart, 1975; Thorndyke, 1975; Schank, 1975) because it is well structured and well known in our culture. The narrative story has an initial situation, which can be broken down irto setting, time, and characters; a complication which should be interesting and which causes actions or reactions by characters; a resolution to the complication; and an optional evaluation and/or moral. There must be at least one episode (sequence of initial situation, which may be the outcome of a previous episode, complication and resolution), but there may be more. When a person reads a narrative,

the slots of this schema are filled in with story-specific information which satisfies the conditions of each particular branch. This story-specific information corresponds to the most general level of the story.

The lowest level of a discourse is called the microstructure. This level is formed from the text itself. As one reads (or hears) information, this information is organized into idea units, called propositions. In addition to those propositions derived directly from the text, other propositions can be inferred from the text information or supplied from previous knowledge about the topic.

Through application of operations such as deletion of irrelevant details, abstraction, and transferral of important propositions intact, this level can be transformed into a more general one called the macrostructure. These operations can be used recursively to obtain new macrostructures at increasingly more general levels of information.

The schema directs the formation of macrostructure by applying the operations to a particular level of propositions so as to reduce and organize the information in the text. In this process, information in the schema is utilized to ensure that important information is not deleted and irrelevant or inferrable information is not retained. Many of the microstructure propositions are lost (forgotten) when they are eliminated from inclusion in the macrostructure. The final level of macrostructure contains the information which fills in the slots at the terminal nodes of the schema.

While the schema is an abstract knowledge structure, the macrostructure is derived from a particular text. The schema is not directly observable. It can be inferred, however, from the similarity in how people treat a well known class of information, such as narrative stories and from the incomprehensibility of information for which no schema is available. The effects of the narrative can be observed by examining recall protocols or summaries for macrostructure propositions.

Much of the information available in the recall paradigm is not directly relevant to the study of macrostructure at a particular level. This is because when asked to recall a text, individuals typically include a large amount of high-level information from the macrostructure. In addition, they recall idiosyncratic details from the microstructure.

A second method used to examine macrostructure is summarization. A summary is a text based on the most important propositions of another text. These are the macro-propositions. By eliminating much of the detailed information in the text, summaries tend to reflect only the macrostructure propositions.

A number of levels of macro-propositions are possible since the operations which form macrostructure can be applied recursively. In order to compare the summaries of different subjects, it is necessary to ensure that the level of detail (macrostructure) in the summaries is not radically different. Therefore, to enable comparability, summaries are restricted to a maximum and minimum number of words. Because of this restriction, all participants tend to use macrostructure propositions at a similar level.

Kintsch, Kozminsky, Streby, McKoon, & Keenan (1976) used the recall paradigm to study the macrostructures underlying comprehension of history and science paragraphs. Recall was compared to theoretically derived hierarchical structures for each paragraph. They found that participants tended to recall information far more frequently from higher levels of the hierarchy than they did from the lowest levels. This is consistent with the hypothesis that micro-level propositions tend to be forgotten while macro-level propositions are stored.

If a schema is not available for a text, the memory for the text is retarded. What is recalled is more idiosyncratic than recall for a text for which the reader has a schema. Kintch and Greene (1978) used a sequential recall task to examine the effects of the availability

of a schema on comprehension and recall using stories with familiar and unfamiliar schemata. Using a series of five participants, with each participant re-telling the story to the next, the final version of an Apache Indian tale was very short and idiosyncractic. A Decameron story, however, was intact and showed much better recall. Participants in this experiment were familiar with western-culture literature so that appropriate, well-developed schemata were available for the Decameron story; this is, of course, not true for the Indian story which is from a different culture. Although these participants were "experts" in comprehending one type of story, they were "novices" in comprehending the other.

Kintsch and Kozminsky (1977) utilized the summarization technique to examine differences between reading and listening. A comparison of the summaries showed remarkable consistency among groups. Propositions in the upper levels were recalled by participants more frequently.

As demonstrated by Kintsch and van Dijk (1978), theoretically derived macrostructures predict quite well the propositions included in participants' summaries. Although the research cited above is primarily concerned with how newly presented information is understood, we feel that it is also relevant to examining existing knowledge structures.

Overview of the Current Research

In our review of the literature on expert and novice problem solvers, we concluded that expert problem solving behavior is strongly influenced by existing memory structures or schemata. A similar result is also apparent in our discussion of the literature on text comprehension (e.g., Kintsch and Greene, 1978). There are many similarities between the schemata involved in text processing and the hierarchical knowledge structures employed by expert problem solvers.

The concept of schemata can also be used to differentiate the knowledge structures used by experts and novices in a given problem solving domain, such as comprehension of software design problems.

The research reported here is a preliminary study to examine the knowledge structures used by experts and novices in comprehending software designs. In order to examine the knowledge structures used in software design, we have attempted to elicit the subjects' macrostructures. Recall that the macrostructure of a text is more directly observable than the schema used to organize and comprehend the text. As we indicated above, the experimental paradigm of summarization is more useful then a recall task to examine macrostructures because less extraneous information is included that is not part of the macrostructure in summaries than in recall. In the present study, we have used this technique to assess differences in the knowledge structures employed by subjects with different levels of experience in software design.

We will now briefly consider the types of results that we would anticipate from the application of this technique to the analysis of the knowledge structures involved in software design, or any other problem solving task. First, we would expect a summary, which in the experiment reported here is a summary of the specifications for a computer system, to include that information that a participant considers to be "most important". We argue that "most important" is actually a function of a participant's experience. In summarizing a familiar narrative, for example, a participant provides the macrostructure of this particular narrative; the participant does not provide information such as the fact that the narrative involves an initial situation, a complication, a resolution, etc. Since a participant expects to see this type of information and since these expectations are confirmed, the participant does not consciously attend to this information. That is, what is "obvious" is not considered important or essential to processing the text.

If we consider now a complete macrostructure that underlies the specifications of a software system, we would expect relatively inexperienced participants to concentrate on the higher levels of this structure while more experienced participants would ignore this "obvious" information and concentrate on the more important information at the lower levels. This obvious information is already integrated into the knowledge structures used by the experienced participant in this task.

In Figure 1, we present a prototypical macrostructure for a software design. At the top level, we assume that a design can be considered to consist of "inputs", "processes", and "outputs". At the lower levels, each of these elements is further decomposed or refined. As a person gains experience, memory structures are developed that encompass more and more levels of this structure. We would predict, therefore, that the novice would concentrate on the higher levels while people with more experience would concentrate on the lower levels in reading and summarization. By comparing the summaries provided by participants at different levels of experience, we should be able to develop insights into the nature and evolution of the memory structure, or schemata, involved in software design.

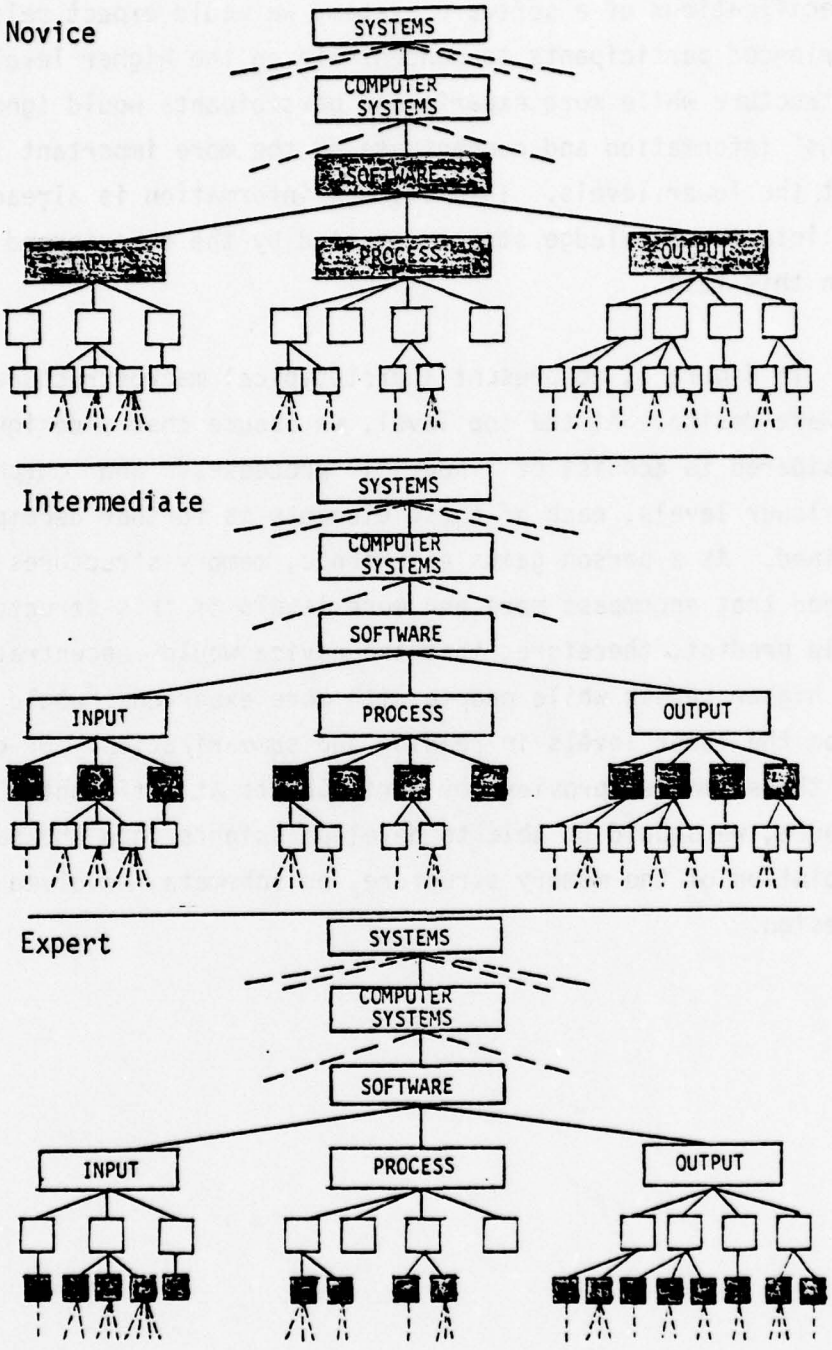


Figure 1

A generalized representation of a software design structure.

(Shaded areas show top level(s) of macrostructure.
Area above shaded portion is inferred schema.)

METHOD

Participants. Thirty undergraduate students at the University of Colorado (Boulder campus) were recruited through newspaper advertisements and classroom announcements. Participants were paid for participating in the experiment. Six participants were assigned to each of five experimental groups on the basis of formal coursework in computer science. Participants in the first group had no courses. Those in Group 2 were currently enrolled in their first course and those in Group 3 were currently enrolled in their second course. These two courses comprise an introductory sequence and are required for further courses. Group 4 participants had completed three or four formal courses and those in Group 5 had completed five or more courses.

Apparatus. The execution of this experiment was controlled by a Xerox Sigma 3 computer. Participants typed their responses on a keyboard that was connected to a IV-Phase System CRT Display Terminal.

From one to six participants were run concurrently under the control of the real-time computer system. The procedure was participant-paced and an independent sequence of events was presented to each participant. Terminals were arranged in pairs and each pair was in a small room off a large common room. The display terminals were on a 1.2 x 0.75 m table. The terminals in each experimental room were positioned so that participants faced opposite walls.

Materials. Material consisted of a description of the functional requirements of a computer system to process telegrams. This text was typed on a single page and labelled "TELEGRAM PROCESSOR" at the top. This text was further divided into a short background section, labelled "Background" and a description of the specifications, labelled "Design Task". The entire text that was presented to participants was approximately 270 words in length and is presented in Figure 2.

TELEGRAM PROCESSOR

Background

A system is required to process a stream of telegrams. Hardware consists of a batch processing system and includes a line printer and a paper tape reader.

Design Task.

You are to design a system that will process a stream of telegrams. This stream is available on paper tape as a stream of letters, digits, and blanks.

The tape is accessed by a "read block" instruction which reads into main storage a variable length character string delimited by a terminal EOB (End of Block) character. The size of a block cannot exceed 100 characters, excluding the EOB characters.

The words in the telegram are separated by sequences of blanks and each telegram is terminated by the occurence of the word 'ZZZZ'. The stream is terminated by the occurence of an empty telegram -- that is, a telegram with no words, followed by 'ZZZZ'.

Each telegram is to be processed to determine the number of chargeable words, and to check for occurrences of overlength words. The words 'ZZZZ' and 'stop' are not chargeable and an overcharge is added if one or more words exceed 12 characters in length.

Telegrams are to be printed on a line printer. When possible, each output line should be between 100-120 characters in length. No word should be split between the end of one line and the start of the next. You may assume that all words will contain fewer than 20 characters.

Finally, extra blanks in the telegram are to be deleted on output and the word count and an overcharge message (if necessary) are to be printed after each telegram.

Figure 2.

Problem Statement

Procedure. Participants were given a typewritten copy of Telegram Processor text with instructions to read it for comprehension and to make sure that they understood it. The time that each participant took to read and understand this text was recorded. When participants indicated that they understood the problem statement, each was asked to write two summaries. One was a summary of the design specifications that the participant had just read. The other was a summary of what the design for the indicated computer system would look like, if each participant actually were to complete the design. This was done in order to determine if there were any significant differences in the type of information contained in these two types of summary. At no time, however, were participants actually required to perform the design task.

The order of writing these summaries was counterbalanced so that half the participants in each experience group did the specification summary first and half did the design summary first. The text containing the design specifications was available to subjects while they were writing both types of summaries. In this way, any effects due to memory limitations were eliminated, as the main interest in this study was the structure imposed on the text by participants of differing levels of expertise.

RESULTS

Three different types of analyses were performed on the data. The first considered surface features of summaries, such as reading times, writing times, and number of words included in summaries. The second focused on the number of ideas, or propositions, contained in the specification and design summaries. The final type of analysis concerned the type, or content, of propositions contained in the summaries.

Surface Measures

The first set of measures that we will report can be characterized as "surface" information. This information is concerned more with surface properties of subjects' responses than with the ideas conveyed in them. Two of these are temporal measures - reading/study time (or simply reading time) and writing time. The last measure to be reported is word count.

The reading time data for five participants in Group 2 and one each in Groups 3 and 4 was lost when these participants accidently erased this information. Since there was only one score for Group 2, this group was eliminated from the analysis of these data. No differences were reliable among the remaining four groups (see Table 1). There is a tendency in the data, however, for times in Group 3 to be somewhat slower than those of the other three groups.

Due to the large variability in the data, there were no reliable differences for experience level in writing times for either specification or design summaries (see Tables 2 and 3). Participants who wrote specification summaries first, however, wrote longer than participants who wrote them second ($F(1, 20) = 5.631$, $p < .05$).

1	2	3	4	5	Overall
263.67 (99.03)	406.00* (***)	385.60** (160.72)	279.80** (98.28)	232.83 (76.63)	291.826 (117.50)

* one score only
** five scores

Table 1

Means and (Standard Deviations)

Reading Times (in seconds)

	Experience Level					
	1	2	3	4	5	Overall
1*	3300.00 (1526.44)	2663.67 (2172.05)	3270.67 (1794.50)	2144.33 (1014.64)	995.33 (647.35)	2474.80 (1567.01)
2*	2237.33 (1205.22)	1698.67 (809.34)	1048.67 (397.83)	1194.33 (251.36)	1024.00 (447.40)	1440.60 (771.18)
Overall	2768.67 (1360.81)	2181.17 (1558.37)	2159.67 (1683.03)	1669.33 (841.32)	1009.67 (497.93)	1957.7 (1322.55)

* 1 indicates specification summary first; 2 indicates specification summary second.

Table 2

Means and (Standard Deviations)
of Writing Times for
Specification Summaries (in seconds)

	EXPERIENCE LEVEL					
	1	2	3	4	5	Overall
1*	1386 (200.33)	666.33 (560.12)	1425.67 (82.53)	1761.00 (64.82)	1308.00 (309.11)	1376.07 (292.03)
2*	1906.00 (1428.40)	1389.00 (160.78)	2669.33 (2508.61)	2296.33 (900.63)	1408.50 (570.64)	1971.36 (1304.10)
Overall	1646.00 (955.67)	1027.67 (540.84)	2047.5 (1727.42)	2028.67 (641.95)	1348.20 (363.61)	1663.45 (961.19)

* 1 indicates design summary first; 2 indicates design summary second.

Table 3

Means and (Standard Deviations)
of Writing Times for
Design Summaries (in seconds)

As expected, since summary lengths were limited, there were no reliable differences in the number of words observed for specification summaries or for design summaries. The order of the tasks also had no significant effect on the number of words written for either summary type (see Tables 4 and 5).

To summarize, the analyses on surface features of the data--reading times, writing times and number of words in specification and design summaries -- did not show any reliable differences among the experience groups. The order of the summary tasks had some effect on the writing times for specification summaries but not for design summaries. Order did not interact with experience on any surface measure.

Analyses of Propositional Ideas

The remaining analyses to be reported here are concerned with the ideas which are expressed in the summaries. To assess these ideas and their relationship to those in the text, we have adapted a method of propositional analysis proposed by Kintsch (1972, 1974) and developed by Turner & Greene (1978). Propositional analysis provides an objective means for analyzing the "idea units" or propositions in a text.

This method was modified to enable the scoring of each protocol for its propositional content without resorting to too much detail. The micro-level propositions were grouped into clusters, generally involving a predicate proposition and its modifiers. (A predicate proposition is one that involves a state or an action. A modifier proposition can express qualification, quantification or circumstance.) To clarify this distinction, consider an example. One of the statements in the "Telegram Processor" text is "The words in the telegram are separated by sequences of blanks ...". Using Kintsch's method, we obtain three micro-level propositions (Table 6).

	EXPERIENCE LEVEL					
	1	2	3	4	5	Overall
1*	78.00 (1.00)	77.67 (2.52)	81.33 (2.52)	77.33 (11.93)	71.33 (8.33)	77.13 (6.74)
2*	77.00 (8.72)	78.67 (2.31)	71.67 (8.74)	66.33 (7.55)	70.33 (8.96)	72.80 (7.48)
Overall	77.50 (5.58)	78.17 (2.23)	76.50 (7.82)	71.83 (9.15)	70.83 (7.76)	74.965 (7.15)

* 1 indicates specification summary first; 2 indicates specification summary second.

Table 4

Means and (Standard Deviations)
of Number of Words in
Specification Summaries

	EXPERIENCE LEVEL					
	1	2	3	4	5	Overall
1*	71.33 (7.51)	73.33 (11.55)	72.33 (10.02)	73.33 (9.87)	76.67 (3.51)	73.40 (7.78)
2*	66.33 (2.39)	69.67 (8.15)	73.33 (10.69)	80.00 (0.00)	70.33 (7.23)	71.93 (7.79)
Overall	68.83 (6.56)	71.50 (9.16)	72.83 (9.28)	76.67 (7.23)	73.50 (6.16)	72.66 (7.685)

* 1 indicates design summary second; 2 indicates design summary first.

Table 5

Means and (Standard Deviations)
of Number of Words in
Design Summaries

For our purposes, γ is the only proposition scored because it is a predicate proposition. The other propositions represent refinements of this basic idea.

Using this modification, we identified 27 cluster propositions in the text. These 27 propositions were placed in a tree structure based on their level within a hierarchical net of subordinate relationships. The proposition clusters and the hierarchy can be seen in Figure 3.

Each summary was scored for total number of proposition clusters. This includes correct and incorrect propositions, inferences, elaborations, etc. In addition, each specification summary was scored a second time to count those propositions that correctly reproduce propositions which are included on the list of 27 cluster propositions in Figure 3. These will be referred to as "reproductive" propositions. For example, if a participant were to include in his or her summary the statement, "Output each telegram to the printer," this participant would receive credit for reproducing Proposition 21 ("Telegrams are printed on the line printer").

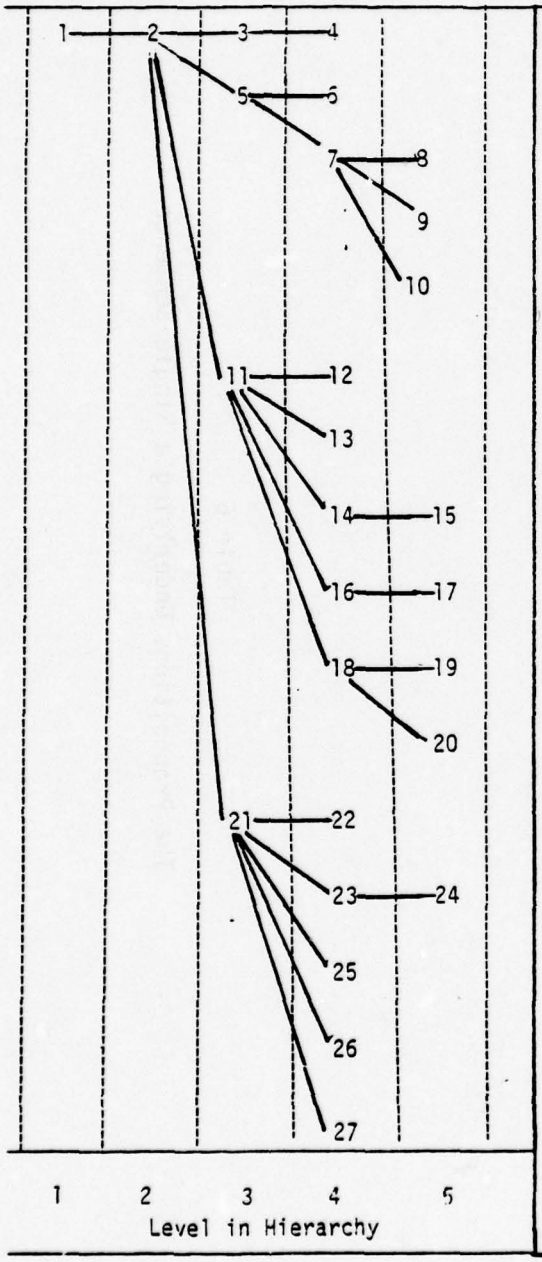
Design summaries were scored a second time as well. This analysis included any reproductive propositions plus any propositions which were determined to be pertinent to a design by an expert judge. These "pertinent" propositions which are correct statements, may be ideas that are inferred from or abstracted from the specifications of the design problem or supplied from participants' existing knowledge structure. This set of propositions -- reproductive (as defined above) and pertinent -- will be referred to as "relevant" propositions. An example of a pertinent proposition would be "Calculate the cost of each telegram, which is cost per word times number of words plus overcharge". At no time in the specification was a proposition such as "calculate the cost of each telegram" included. Yet this is a relevant and correct idea for the design of this system.

Identifier	Proposition	Meaning in Sentence	Type of Proposition
α	(LOCATION: IN, WORDS, TELEGRAM)	Words are in telegrams.	Circumstance
β	(CONSIST OF, SEQUENCES, BLANKS)	There are sequences of blanks.	Qualification
γ	(SEPARATE, β α)	Words are separated by blanks.	Predication

Propositional representation of the sentence
"The words in the telegrams are separated by sequences of blanks."
using Kintsch's method of propositional analysis

Table 6

The Propositions Underlying a Single Sentence



1. Design a system.
2. System processes telegrams.
3. Hardware consists of batch processing system.
4. (3) includes line printer and paper tape reader.
5. Stream of telegrams available on paper tape.
6. Stream consists of letter, digits and blanks.
7. Access input by "Read Block".
8. "Read Block: reads a variable length character string.
9. Block terminated by EOB character.
10. Block $\leq$ 100 characters.
11. Process each telegram
12. Words are separated by blanks.
13. Each telegram terminated by "ZZZZ".
14. Each stream of telegrams terminated by empty telegrams.
15. Empty telegram - telegram with words followed by "ZZZZ".
16. Determine number of chargeable words.
17. Words "ZZZZ" and "STOP" not chargeable.
18. Check for overlength words.
19. Overcharge added if (20).
20. Overlength > 12 characters.
21. Telegrams are printed on line printer.
22. Output line is between 100-120 characters.
23. Do not split words.
24. Words < 20 characters.
25. Delete extra blanks on output.
26. Output word count.
27. Output overcharge messages.

Figure 3

Hierarchical Structure of Telegram Processor Text

The total number of idea-level proposition clusters, which includes both relevant and irrelevant clusters, did not differ among experience groups for either specification or design summaries. Order also had no effect, and there was no significant interaction between order and experience level. Controlling the number of words written in a summary effectively controlled the number of clusters included in the summaries.

The number of reproduced proposition clusters for specification summaries did not differ significantly among experience groups, nor did order of task have an effect.

As noted above, the analysis of design summaries considers all relevant (reproductive and pertinent) propositions. The results of a two-way analysis of variance show that experience is not significant at the .05 level of confidence ($F(F4, 20) = 2.6547$, $p = .063$). However, this F-ratio approaches significance due to the large difference between naive participants and those in the other four groups (see Table 7). Design summaries for naive participants contain fewer relevant proposition clusters than those of the other four groups. Half of the naive participants included no relevant propositions in their design summaries. There were no differences among the other four groups in the amounts of information included. There are no significant effects for order of task or the interaction of experience and order.

To summarize, the total number of propositions, or the number of propositions which reproduce the text or are relevant to the design, are not significantly affected by experience or by the order of tasks.

Analyses of Propositional Type

The next set of analyses takes into account what is written in the summaries, not just how much is written. Three separate divisions

EXPERIENCE LEVEL

1	2	3	4	5	Overall
4.17	9.17	10.17	10.67	9.83	8.80
(5.95)	(3.06)	(2.40)	(1.75)	(5.27)	(4.45)

Table 7

Means and (Standard Deviations)
of Number of Relevant Proposition
Clusters in Design Summaries

of the proposition list for the text were made. The first analysis divides propositions into three categories -- Input, Process, and Output. The second analysis breaks the propositions down into five hierarchical levels and the final analysis focuses on the level of detail included in the summaries.

In performing these analyses, we elected to redefine the experience group factor. Group 1 was the "Low Experience" group and was composed of students with no computer science experience, Groups 2 and 3 were combined to yield a "Medium Experience" group. These students, who were in the process of taking the required introductory sequence in computer science, are just learning the basic concepts. Groups 4 and 5 were combined to form a "High Experience" group. These students have taken upper level courses involving more in-depth concepts. This regrouping enabled us to more easily examine trends involving the kinds of information that participants at differing levels of expertise attend to.

In the first analysis, propositions were divided into three categories -- Input, Process and Output. "Input" propositions include proposition 5 ("the stream of telegrams in on paper tape") and its subordinates (6, 7, 8, 9, 10; see Figure 3). "Process" propositions include Proposition 11 ("Process each telegram") and all its subordinates. "Output" propositions are Proposition 22 ("Telegrams are printed on line printer") and those below it. Propositions 1 through 4 were excluded from this analysis.

Summaries were examined in terms of the percent of Input, Process, and Output information which was reproduced. Reproductive specification propositions and relevant design propositions were classified on this dimension. Several propositions did not fit into this framework. These propositions were either macro-level propositions ("Design a system", "the system should process streams of telegrams") or propositions related to hardware. These were excluded from the present analysis.

Analyses were performed separately for Input, Process, and Output information (see Figure 4). In all cases, participants included more information in their specification summaries than in their design summaries, regardless of level of experience. This factor, summary type, was significant in each analysis. (Input Information: $F(1, 54) = 5.9299$, $p < .05$; Process: $F(1, 54) = 5.0393$, $p < .05$; Output: $F(1, 54) - 8,4569$, $p < .01$). In addition, a higher percentage of process information was included by Medium and High experience participants than by novices. ($F(2, 54) = 8.8475$, $p < .001$). Interaction between Summary Type and Experience Level was not significant.

The second propositional-type analysis divided the propositions of the problem statement into five hierarchical levels as indicated in Figure 3. Level 1, represented by a single proposition ("Design a system") is the superordinate proposition in this text. Level 2 is also a single proposition ("The system is to process a stream of telegrams"). Level 3 breaks the problem down into four areas - Hardware, Input, Process, Output. Levels 4 and 5 are further details of the system's specification.

Each participant's summary was scored for the percent of the propositions reproduced on each level. The low experience group included far more Level 2 and 3 propositions in their summaries than did more experienced groups (see Figure 5). Participants in all three groups included similar amounts from Level 4. Level 5 propositions were included most frequently by the high experience group, slightly less often by the medium group and very infrequently by the low experience group (see Figure 5).

A two-way unequal-n analysis of variance indicated that the Level in the Hierarchy had a reliable effect on inclusion of propositions in summaries ($F(4, 135) = 2.8905$, $p = .025$). While Experience is not a significant factor, the interaction of Hierarchy and Experience is significant ($F(8, 135) = 2.3186$, $< .05$).

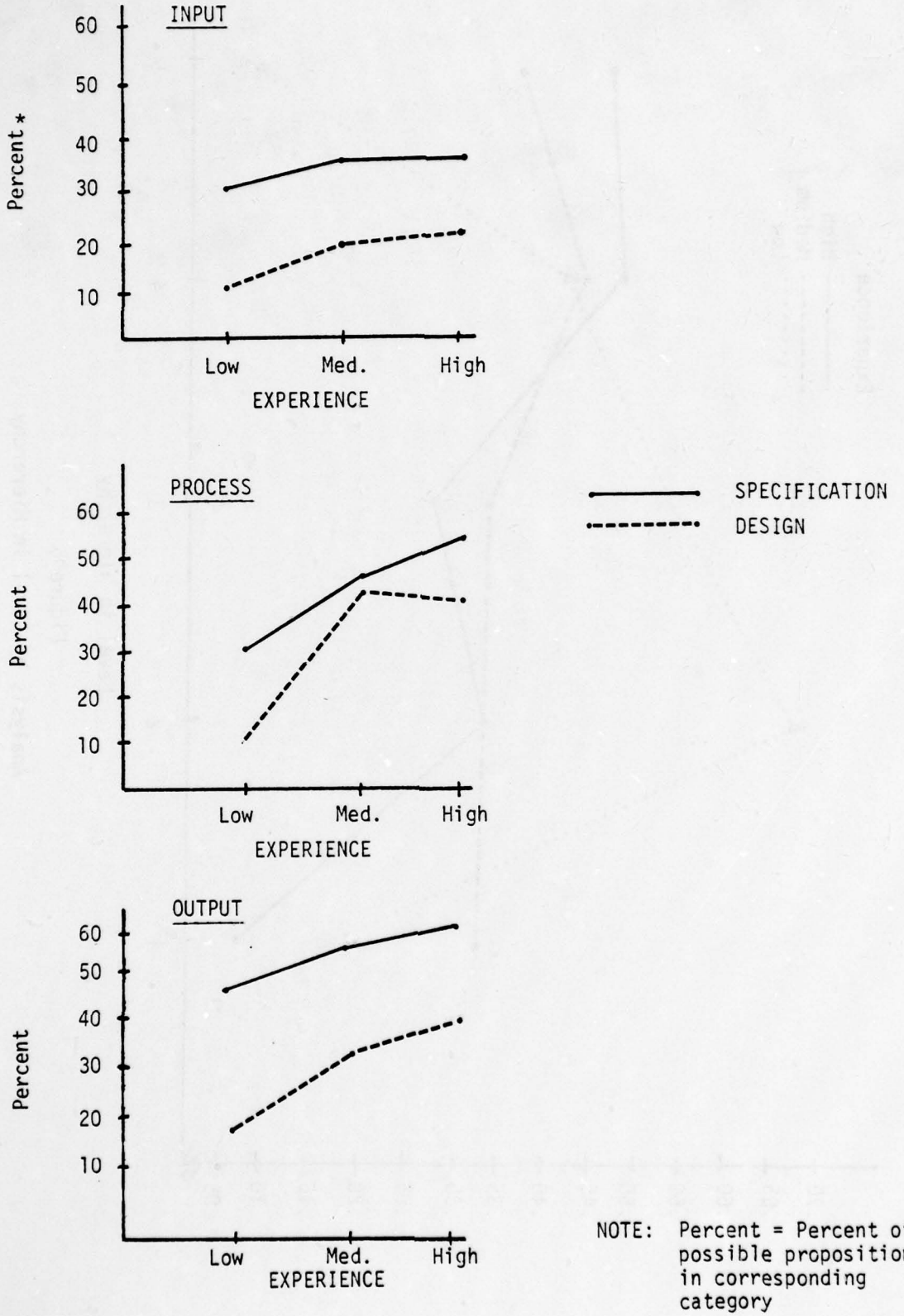


Figure 4

Input-Process-Output Analysis

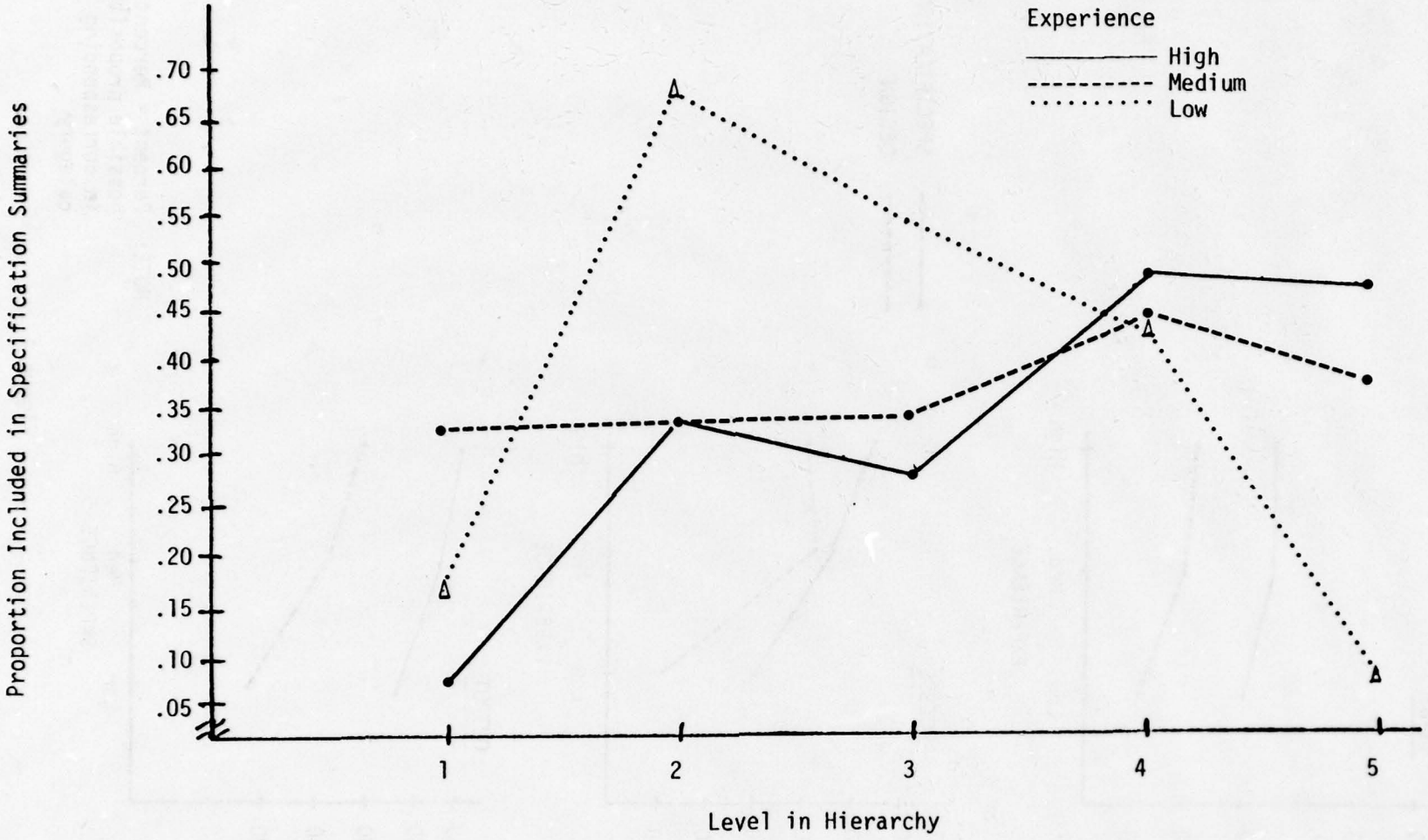


Figure 5

Analysis by Level in Hierarchy

The last analysis focuses on the level of detail included in the summaries. For this measure, propositions on the terminal nodes in the hierarchical structure were considered the lowest level of detail in the specifications. Thus, any proposition which had other propositions subordinated to it was considered a "general" proposition. General propositions are #1, 2, 3, 5, 7, 11, 14, 16 18, 21 and 23. All other propositions were considered "detail" propositions. A percent score for each type was calculated for each participant.

Low experience participants included mostly general propositions in their specification summaries (see Figure 6). High experience participants included far more detail than general propositions. The medium group included roughly the same amount of each. A two-way unequal-n analysis of variance indicated that the two main effects, experience and level of detail, were not significant. However, the interaction of Experience with Generality was highly significant ($F(2, 54) = 5.5358$, $p < .01$).

In summary, the amount of Input-Process-Output information included in specification and design summaries was significantly different, but this variable did not interact with Experience. The level in the hierarchical structure of the text was a significant factor and interacted with experience. The interaction indicated more lower-level information in experienced participants' summaries, while more upper-level information appeared in inexperienced participants' summaries. The Generality-of-Propositions dimension interacted with Experience although neither factor was significant alone. Experienced participants concentrated on detailed information, while inexperienced wrote more general information. This result is illustrated in Figure 7. In the figure, those propositions used by at least half the participants in a given group are circled.

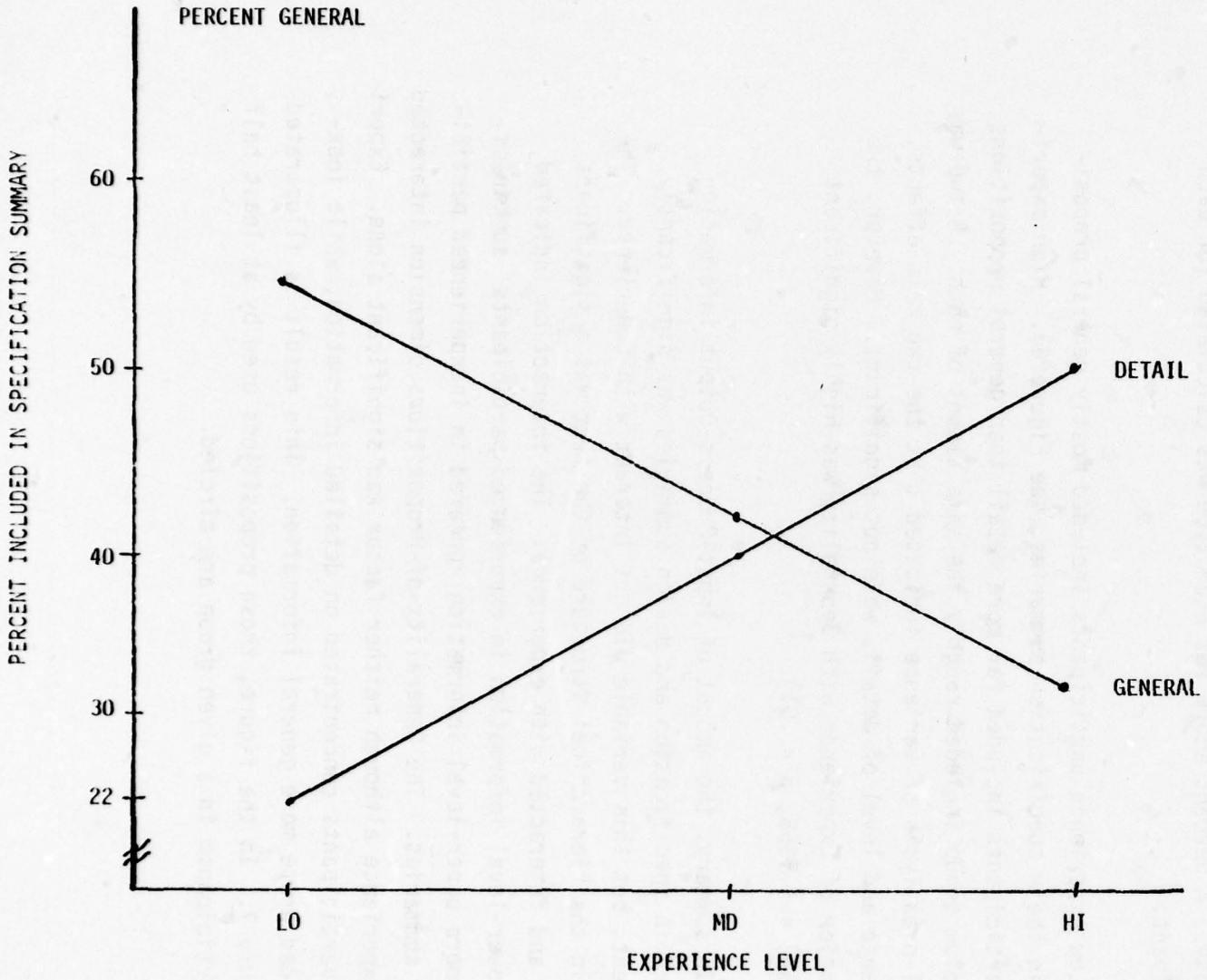


Figure 6
Analysis of General and Detail Information

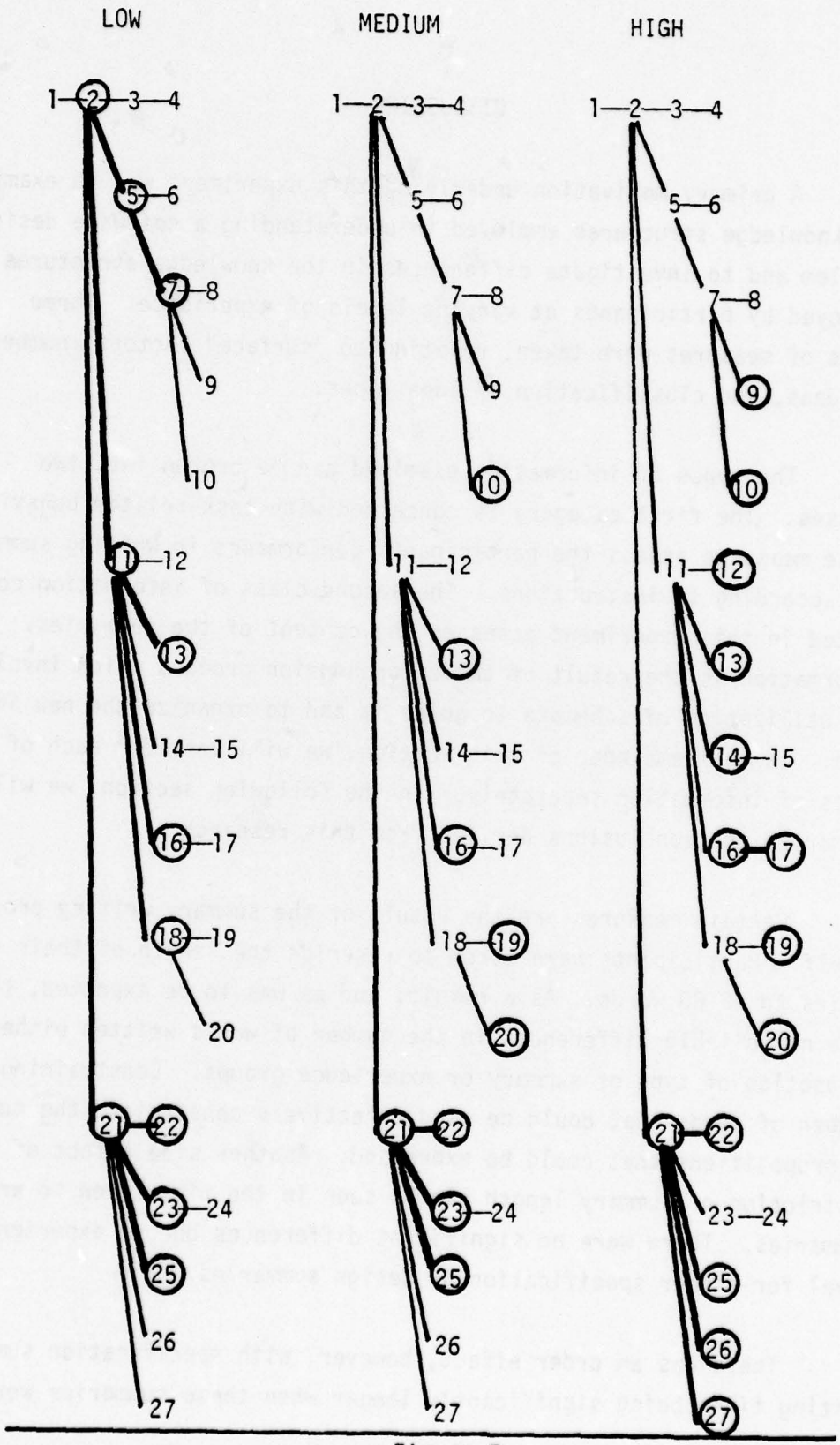


Figure 7

Proposition Usage as a Function of Experience Group

DISCUSSION

A primary motivation underlying this experiment was to examine the knowledge structures employed in understanding a software design problem and to investigate differences in the knowledge structures employed by participants at varying levels of experience. Three types of measures were taken, relating to "surface" factors, number of ideas, and classification of idea types.

The types of information examined can be broken into two classes. The first category is concerned with task-related behavior. These measures assess the participants performance in writing summaries according to instructions. The second class of information collected in this experiment assesses the content of the summaries. This information is the result of the comprehension process which involves the utilization of schemata to guide it and to organize the new information. In the remainder of this section, we will consider each of these types of information separately. In the following section, we will summarize the conclusions derived from this research.

Certain measures are the result of the summary writing process itself. Participants were asked to restrict the length of their summaries to 60-80 words. As a result, and as was to be expected, there were no reliable differences in the number of words written either as a function of type of summary or experience groups. Constraining the number of words that could be used effectively constrained the number of propositions that could be expressed. Another side effect of the restriction of summary length can be seen in the time taken to write summaries. There were no significant differences due to experience level for either specification or design summaries.

There was an order effect, however, with specification summary writing times being significantly longer when these summaries were

written first than when they were written after the design summaries. Since it is reasonable to expect that participants would require some amount of learning time to be accustomed to the experimental apparatus, we would expect that the time required to write the first summary, regardless of type, would be somewhat longer than the time required to write the second. We have no explanation for the fact that only specification summary writing times were so affected. Additional research would be required to determine if this is a reliable result or is due primarily to the small sample sizes and large observed variances in the present experiment.

The number of reproductive propositions included in a specification summary was also affected by the restriction on summary length. There was no difference in the observed number of reproductive propositions due to experience level. Recall that reproductive propositions were defined as those that were derived directly from the information stated in the problem statement. Since the problem statement was available to participants while they were writing their summaries, memory effects due to differences in comprehension level and organization are unlikely. Thus, the task for the specification summary was one of <u>selection</u> of information from a known set.

To some extent, reading time is also a function of the experimental task. Since participants knew that the problem specification would be available to them during their summarizing tasks, they were not necessarily motivated to comprehend the problem at a deep level during the reading and studying part of the task. No reliable differences in reading/study time due to experience level were observed. A difference in reading/study time, had such been observed could indicate that participants with different levels of experience were doing different things as they read the text.

Reading time, however, is not entirely a function of the experimental setting, but can also be influenced by the schemata used

by participants to process the text. If this were the case, the effects of experience on reading time would produce a U-shaped function.

Recall that participants in Group 1 had no previous computer-relevant experience. Because of their unfamiliarity with the computer-related terms and concepts expressed in the problem statement, we would expect these participants to stop studying the problem statement when they had only a very superficial grasp of its contents. That is, although these participants could read the problem statement as "text", their lack of relevant experience and corresponding appropriate memory structures, or schemata, prevented their actually understanding the problem statement at anything other than a superficial level. Reading times for Group 1, therefore, should be fairly low.

Reading times for Groups 4 and 5, who were the most experienced participants, should also be fairly low. In this case, their experience allows them to "skim over" or ignore information that they consider to be irrelevant to solving the stated problems and to focus on information that they consider more important. In this case, low reading times are attributed to the presence, rather than absence, of appropriate schemata.

Participants in Groups 2 and 3, on the other hand, were more familiar with the computer-related concepts than Group 1 participants but lacked the experience of those in Groups 4 and 5. Although these participants understood the individual concepts, they have had little, if any, experience in integrating these concepts to formulate a design for a computer system or group. Attempting to integrate these concepts, therefore, takes a significant amount of time and effort and results in fairly long reading times.

Reading times, therefore, are affected both by the experimental setting and the schemata used to process the problem statement.

Although the effects of experience level failed to reach significance (at the .05 level of confidence), an examination of Table 1 suggests that the pattern of reading times were, to a degree, influenced by schemata.

Unlike the measures discussed above, the remaining measures are primarily concerned with a participant's understanding and organization of the problem specifications, in particular, and computer systems, in general. This understanding, we believe, is structured by a participant's schemata, which are largely determined by experience.

The schemata used by a naive participant in this task is probably a very general schema for processing a text. In addition, naive participants should have difficulty in understanding the text as a design problem because of their lack of knowledge about computer systems and designs. Participants in all other groups should have schemata for computer science problems. The more experienced a participant is, the more detailed his or her schema should be. These participants should be able to generate propositions which are relevant to a design.

The propositional content of the summaries is interpreted as a macrostructure which is produced by the interaction of an individuals' schemata with the text. Differences in the ideas included in the summaries represent differences in the schemata utilized in the comprehension and organization of the problem. The schemata are not directly observable in the summaries since people rarely include information from their schemata in summaries, because it is too "obvious". However, summaries should include the top levels of macrostructure immediately below the level of schemata. Therefore, the schemata for a given experience group are inferrable from the macrostructure propositions that they include.

In order to familiarize the reader with the nature of the summaries, we present prototypical summaries in Figures 8 and 9 for specifications and designs, respectively. With the exception of the design summary for the low experience group, the illustrated summaries contain those ideas stated by at least half of the participants in each group. Since there was no such concensus in the design summaries of the low experience group, a representative summary is presented.

Without a schema for a software design, naive participants asked to do design summaries have difficulty determining what information is relevant to the problem at hand. For experienced participants, the design summary task entails inclusion of information that is not directly stated in the problem as well as information that is reproductive.

In the analysis of the design summaries, we judged propositions on the basis of relevance to software design; that is, propositions were judged on the basis of whether or not the information conveyed was, in fact, related to or consistent with a software design task. As would be expected, there was a tendency for more experienced participants to achieve higher relevancy scores, although this effect was not significant at the .05 level of confidence. Extremely low scores attained by the naive (Group 1) participants, can be attributed to this group's lack of experience and familiarity with the concepts involved in software design. This suggests that more experienced participants are retrieving and utilizing some schemata or memory structures that contain useful, relevant information about software design.

Some insights into the structure, content, or nature of the knowledge structures that are brought to bear on a software design task can be derived from considering the types of propositions that are used in participants' summaries. Obviously, there are several

Low (Group 1)

This system should process a stream of telegrams. The paper tape is accessed by read block. The end of the telegram would have the letters ZZZZ. Process the telegram to determine the number of chargeable words and to check for overlength words. Telegrams are printed on a lineprinter, each line being between 100-120 characters in length. Words cannot be split from line to next. On output, extra blanks deleted.

Medium (Groups 2 and 3)

The block size cannot exceed 100 characters. ZZZZ will denote the end of a telegram. Count the number of words to be charged and overcharged. There is an overcharge if a word exceeds 12 characters. The telegrams will be printed on a printer with lines of 100-120 characters and no words split between lines. Extra blanks are deleted.

High (Groups 4 and 5)

Blocks are 100 or less characters. The words of a telegram will be separated by blanks, telegrams are separated by "ZZZZ". The stream is terminated by a blank telegram. Count the number of words and check for words longer than 12 characters. The words "ZZZZ" and "stop" are not chargeable. Print telegram on lineprinter, with 100-120 characters per line. Extra blanks deleted. Print word count and overcharge message for each telegram.

Figure 8.
Representative Specification Summaries
for each Experience Level

Low (Group 1)

My first step would be to design a batch processing system, including a line printer and a paper tape reader. The computer would have to complement the functions of the batch processing system. I would set the apparatus up looking somewhat like an assembly line. It would look like the basic computer system, lots of metal, buttons, and complicated gadgetry.

Medium (Groups 2 and 3)

First, check for the end of the telegram. When the end is found, check for blanks. Foe each space increment the number of chargeable words. Check for any overlength words. The words "ZZZZ" and "stop" are not to be counted. Next, calculate the charges for the telegram. Then, output it followed by the word count and the overcharge (if needed). Each output line should be between 100 and 120 characters long.

High (Groups 4 and 5)

Read a block until an EOB character is encountered. Separate the stream into telegrams using the marker "ZZZZ". The next procedure processes each telegram. It counts the number of chargeable words in each one and notes if the character count in a word is greater than twelve characters. The telegram is printed, deleting any extra blanks, followed by the word count and overcharge message. When an empty telegram is encountered, stop processing.

Figure 9.
Representative Design Summaries
for each Experience Level

ways in which types or categories of propositions could be defined. for analysis. Due to the exploratory nature of this experiment and the limited amount of available data, only three such analyses were performed, based on their potential usefulness in detecting differences among groups and for the practical implications that would be implied by such differences. These analyses involve, at a very general level, classification on the basis of computer system function and on the basis of the hierarchical structure underlying the problem statement.

In our first analysis, we partitioned the propositions in the original problem statement on the basis of whether they were primarily concerned with input, process, or output functions. Admittedly, this input-process-output classification is only a gross description of the types of functions that could be involved in a software system. It is, however, a classification that applies to almost all software designs; that is, it has obvious validity and generality. Attempts at more detailed classifications may well lack such generality because they would necessarily be restricted to a limited number of software design tasks. Further, although we argue that this classification is general, notice that all three of these components are not equally emphasized in all designs. That is, it is usually the case that one or another of these components will most affect the overall success of the design effort. It would be expected, therefore, that more experienced subjects would be better able to identify the more crucial element and, as a result, tend to concentrate on this element in their summaries.

Both summaries of the presented specifications and of the ultimate design were included in this analysis. Recall that participants were divided into three experimental groups for the purposes of these analyses in order to highlight group differences and increase statistical power.

For all three analyses (input, process, and output) there was a significant effect due to the type of summary. Since scores were determined on the basis of the percent of the possible propositions in each of the three categories that were included in subjects' summaries and the total number of possible design propositions (reproductive and pertinent) in each category was larger than that for the specification summary propositions (reproductive only), this introduced an artificial difference in the derived scores. This difference, therefore, is considered to be an artifact and is not either theoretical or practical interest.

In terms of the propositions related to the "process" category, there was a significant difference due to experience. As can be seen in Figure 4, there is a general trend for scores to increase as experience increases. This trend can also be noted in the analysis of "output"-related propositions, although this difference did not reach statistical significance. For "input"-related propositions, however, the profile of scores as a function of experience was essentially flat, showing no discernable experience effects.

We interpret these analyses as follows. First, in this particular problem, the design of the input-related processes and structures is relatively simple and straightforward. The process- and output-related processes and structures are more complicated, however. More experienced participants are better able to recognize these potential complications and concentrate on these more relevant issues in their summaries.

Specification summary propositions were also analyzed with respect to their location in the hierarchical structure underlying the original problem statement (see Figure 3). The significant effect due to hierarchical levels was expected. This finding of a significant difference due to levels in the hierarchy tends to confirm that this hierarchy was constructed in a logically consistent manner; failure to detect a significant difference would tend to invalidate the proposed hierarchical configuration.

Of primary interest in this analysis is the significant interaction effect between hierarchical level and experience group. In general, participants in the lowest experience group concentrated on propositions at the highest levels of the hierarchy while more experienced participants tended not to mention these propositions in their summaries and concentrated on the lower-level propositions. As we will argue later, more experienced participants did not include these higher-level propositions because the information contained in these propositions was also contained in the memory structures developed by these participants to deal with software design tasks. In effect, since the information presented corresponded exactly to what participants already knew, participants did not need to explicitly attend to this information.

A related analysis categorized propositions as being either general or detail. This analysis, which is similar to the analysis of hierarchical levels, also produced a significant interaction between experience groups and type of proposition. This interaction is clearly illustrated in Figures 6 and 7. Notice that the amount of general information included in specification summaries declines as a function of experience while the amount of detail information shows a corresponding increase. Again, this suggests that participants do not include information in their summaries that corresponds to already-known information. In effect, participants do not include information that they consider to be "obvious". The type of knowledge structures employed by the more experienced participants is apparent from Figure 7. Those propositions which are high in the hierarchy, but not included in participants' summaries, are presumed to be represented in these knowledge structures, or schemata.

CONCLUSIONS

In this paper, we have focused on the memory structures involved in the comprehension of software system specifications. The primary motivation underlying this research was to determine what these knowledge structures consist of and how they are organized. Although the existence of such knowledge structures, or schemata, has been demonstrated, especially in research on expert versus novice problem solving behavior, the structure and content of these schemata is not well understood.

In an effort to develop such an understanding, we adopted some of the theoretical concepts and experimental paradigms employed in research on text comprehension. This research is largely concerned with the schemata and associated macrostructures that are involved in comprehending and integrating newly presented information. We assumed that there were strong parallels between these types of structures and the types of structures employed by expert problem solvers.

Using a summarization paradigm, the performance of participants with varying levels of experience with software design concepts was compared. Summaries of specification and design information were analyzed. We conjectured that participants at the low end of the experience continuum would concentrate on only the relatively high level concepts. More experienced participants were expected to concentrate on lower level concepts that are more concerned with the details relevant to a successful design or problem solving activity than on the very general concepts involved in this type of activity. As is clearly illustrated in Figure 7, this expectation was confirmed.

By analyzing the information that was included in the less experienced participants' summaries but ommitted from the summaries of more expert participants, the different memory structures employed by the expert and novice were examined.

A large body of research supports the conclusion that expert problem solving behavior is strongly influenced or driven by existing memory structures, or schemata. These schemata contain, in part, generalized plans for solving certain classes of problems. When presented with a given problem, the expert is often able to retrieve one of these schemata and adapt it to achieve a solution to the current problem. In effect, these schemata allow the expert to solve problems more quickly, and frequently achieve better or more appropriate solutions than less experienced subjects. In the experiment reported in this paper, we employed an experimental technique that allowed us to determine the probable content of these schemata, or knowledge structures.

There are several potential implications of this research, but it is necessary to first consider the limitations of the experiment reported in this paper. Since we have reported only an exploratory experiment, there are several limitations that potentially affect the generality of our conclusions. First, we have considered only one software design problem and this problem was relatively simple. Clearly, there are different types of software design tasks and it might well be the case that some tasks, perhaps designing an operating system, are at some level fundamentally different from other tasks, such as designing a system for statistical analyses. In this experiment, we have considered only one of a large class of potential design problems and, correspondingly, only one memory structure that is involved in design. We do not claim that this single structure is general to all software design tasks. In text comprehension research, for example, different schemata are found to underly the comprehension of narratives and articles in a professional journal. We assume, however, that the knowledge structures involved in a variety of software design efforts are similar in structure, although not necessarily in content, and that the principal finding of this study also applies to these as yet unconsidered schemata.

The second limitation concerns the participants that were available for this research. In comparison with professional software development personnel, our most experienced participants were clearly not "expert". It is partially in response to this limitation that we used such a simple design problem. We were able, however, to demonstrate clear effects as a function of experience. We suggest that these effects would be extended with more difficult problems and more experienced participants.

Although we recognize these limitations, we do not feel that they seriously impact the implications of this research. We feel that we have identified a methodology that can extend theoretical research on the nature of expertise in problem solving and have provided some insights into the nature of the knowledge structures underlying expertise. In addition, we have demonstrated the applicability and usefulness of this type of basic research to applied problems.

First, we have demonstrated that a software designer uses previously learned schemata in understanding a software design problem and in actually constructing a design. The actual specifications presented to the designer can vary both in form and content. We suggest that structuring these specifications to more closely correspond to these schemata would aid the designer both in the comprehension of these specifications and the construction of the indicated design.

Second, and of more general consequence, we can develop a much better understanding of the concepts of software complexity and comprehensibility. We have argued that these concepts can only be defined and measured with reference to an individual's experiences and abilities, which are best represented as schemata. By providing a method for determining the structure and content of these schemata, complexity and comprehensibility can now be considered in a more meaningful manner.

As a result of this understanding, we gain a new perspective on relevant metrics of software complexity. Obviously, the development of procedures, techniques, training programs, etc., that lead to the development of more comprehensible software have numerous, significant benefits.

REFERENCES

Bhaskar, R., & Simon, H. A. Problem solving in semantically rich domains: An example from engineering thermodynamics. Cognitive Science, 1977, 1, 193-215.

Chase, W. G., & Simon, H. A. Perception in chess. Cognitive Psychology, 1973, 4, 55-81.

Kintsch, W. Notes on the structure of semantic memory. In E. Tulving & W. Donaldson (Eds.), Organization of Memory. New York: Academic Press, 1972.

Kintsch, W. The representation of meaning in memory. Hillsdale, New Jersey: Erlbaum, 1974.

Kintsch, W., & Greene, E. The role of culture specific schemata in the comprehension and recall of stories. Discourse Processes, 1978, 1, 1-13.

Kintsch, W., Kozminsky, E., Streby, W. J., McKoon, G., & Keenan, J. M. Comprehension and recall of text as a function of content variables. Journal of Verbal Learning and Verbal Behavior, 1975, 14, 196-214.

Kintsch, W., & van Dijk, T. A. Toward a model of text comprehension and production. Psychological Review, 1978, 85, 363-394.

Larkin, J. H. Problem solving in physics (Technical Report). Berkeley, California: University of California, Department of Physics, July 1977.

Miller, L. A. Naive programmer problems with specification of transfer-of-control. AFIPS Conference Proceedings, 1975, 44, 657-663.

Newsted, P. Grade and ability predictions in an introductory programming course (Technical Report). Milwaukee, Wisconsin: University of Wisconsin, School of Business Administration, 1974.

Rumelhart, D. E. Notes on a schema for stories. In D. G. Bobrow & A. Collins (Eds.), Representation and understanding. New York: Academic Press, 1975.

Schank, R. C. SAM -- A story understander (Technical Report 43). New Haven, Connecticut: Yale University, Department of Computer Science, 1975.

Sheppard, S. B. & Love, L. T. A preliminary experiment to test influences on human understanding of software (Technical Report TR-77-388100-1). Arlington, Virginia: General Electric, Information Sciences Programs, June 1977.

Shneiderman, B. Exploring experiments in programmer behavior. International Journal of Computer and Information Sciences, 1976, 5, 123-143.

Sime, M. E., Green, T. R. G., & Guest, D. J. Psychological evaluation of two conditional constructions used in computer languages. International Journal of Man-Machine Studies, 1973, 5, 105-113.

Sime, M. E., Green, T. R. G., & Guest, D. J. Scope marking in computer conditionals -- A psychological evaluation. International Journal of Man-Machine Studies, 1977, 9, 107-118.

Thorndyke, P. W. Cognitive structures in human story comprehension and memory (Unpublished doctoral dissertation). Stanford, California: Stanford University, 1975.

Turner, A., & Greene, E. The construction and use of a propositional text base (Technical Report 63). Boulder, Colorado: University of Colorado, Institute for the Study of Intellectual Behavior, April 1977.

Weissman, L. Psychological complexity of computer programs: An initial experiment (Technical Report CSRG-26). Toronto, Canada: University of Toronto, Computer Science Research Group, 1973.

Weissman, L. M. A methodology for studying the psychological complexity of computer programs (Technical Report CSRG-37). Toronto, Canada: University of Toronto, Computer Systems Research Group, August 1974.